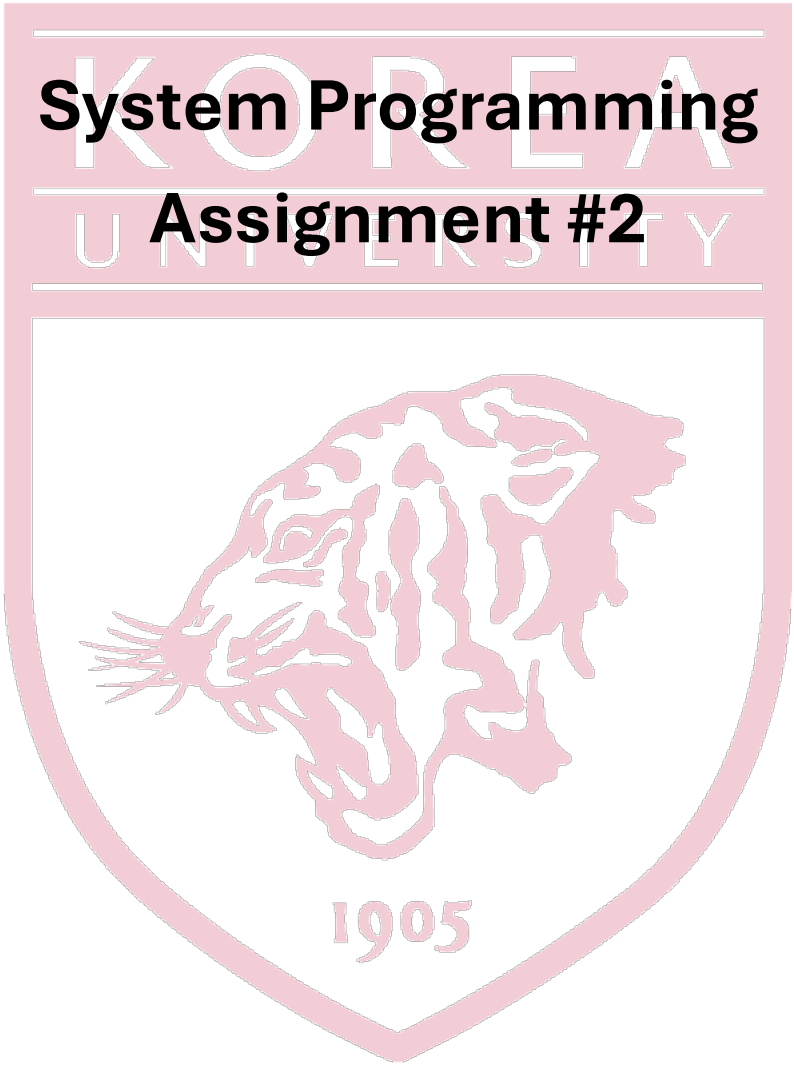
System Programming
Assignment #2



2020320054

박민욱

1. Overall structure of lib.rs

lib.rs는 총 3개의 module로 구성되어 있음.

1 – board_module

2 – block_module

3 – interact_module

첫번째 board_module은 9 x 9 크기의 고정 크기 배열을 활용한 board와 관련된 구조체, associated function, method가 작성된 모듈이고, block_module은 3 x 3 크기의 고정 크기 배열을 활용한 block과 관련된 구조체, associated function, method가 작성된 모듈이며, player enum도 여기에 포함되어 있음. 마지막 module인 interact_module은 command line을 통한 입력과 전체적인 게임 진행을 위한 함수가 정의 되어 있음.

2. board_module

2-1) Board structure

구조체 내부는 9 x 9 2-dimensional array를 사용했고, 배열 내부 data의 자료형은 char로 설정.

2-2) init()

Init()은 Board의 초기 상태로 초기화 하기 위한 일종의 generator로, Board 구조체 내의 data (2-dimensional array)의 요소들을 모두 '_'(underbar)로 통일.

2-3) show()

Show()는 Board structure 내부의 data를 출력하는 부분. 이중 for문을 사용하여 출력함.

2-4) put_possible()

Put_possible()의 경우는 board의 method로, 해당 board의 상태에서 사용될 block과 좌표 (coordinate: (i8,i8)) 가 주어졌을 때 해당 공간에 block이 주어졌는지를 판단하여 boolean 값을 return. 좌표 튜플의 경우 큰 숫자가 필요하지 않기 때문에 i8로 통일하였고, indexing을 위해서 row, col의 경우 usize로 변경. Block을 넣는 것은 3 x 3 크기의 창에서 좌측 상단의 요소를 기준으로 하기 때문에 row, col이 1보다 작거나 7보다 큰, 즉, 유효하지 않은 false를 return. (interact_module에서 이미 이 부분에 대한 처리를 하지만, 혹시 모를 상황을 대비) return할 값은 Boolean 변수 res이며, true로 초기화 후, put할 공간에 이미 값이 존재하는 경우('_'가 아닌 char 값이 들어있는 경우 e.g. @ or 0) res를 false로 변환 후 이중 for문 break. Put_possible은 외부 모듈에서 사용되지 않고, put_block()(2-5), check_lose()(2-6) method에 사용됨.

2-5) put_block()

Put_block()의 경우, put_possible()를 사용하여 해당 공간에 block를 넣을 수 있는지 판단하고 Option을 return하게 됨. 불가능한 경우 None이 return되고, 가능한 경우 이중 for문을 통해 해당 공간에 block을 배치 시킨 후 unit tuple을 감싼 Some 반환.

2-6) check_lose()

Check_lose()의 경우 현재 주어진 block을 기준으로 더 이상 게임이 진행될 수 있는지 없는지를 판단하여 현재 player가 더 이상 게임을 진행할 수 없는 경우 true를 return, 아닌 경우 false를 return하게 됨. 이곳에서 iterator와 closure를 모두 사용하였음. 기본적으로 check_lose()는 사용자의 입력에 의존하는 것이 아니라, 가능한 모든 선택지를 탐색하여 현재 주어진 block을 아무 곳이나 일단 넣을 수 만 있다면 false를 return해야하므로, 탐색하기 위한 좌표를 iterator로 설정. 해당 iterator가 search_coordianate_iter이며, 해당 iterator를 가독성을 유지하며 생성하기 위해 closure를 사용했음.

내부 loop문의 경우, 현재 block을 모든 선택지, 즉 (1,1)~(7,7)까지 모두 순회하며 넣어보고, 넣을 수 있는 경우가 생기면 외부 res를 false로 바꾸고(lose condition이 아님을 의미) 'outer for문을 break. 만일 iterator를 모두 순회하였다면, 내부 loop문을 break. Outer for문이 존재하는 이유는 해당 블록의 rotation도 고려해야함. 같은 모양의 block을 rotate해서 나올 수 있는 경우는 최대 4개이므로 for문은 1..5, 총 4번 반복하게 끔 설정되어 있기 때문에 이중 반복문으로 고려하는 전체 경우의 수는 7 x 7 x 4로 사용자의 입력에 상관없이 게임 승패 여부를 판단할 수 있음.

3. block_module

3-1) Player enum

Player의 경우 2명 뿐이므로 P1, P2로 표현.

3-2) Block structure

Block 구조체의 경우 Board 구조체와 차이는 단순히 크기 만 다른 것. (3 x 3).

3-3) get_block()

get_block()의 경우 현재 player enum과 block_num을 받아서 적합한 block을 반환하는 함수. Player1의 경우 element를 '0', 2의 경우 '@'로 설정하고 element를 이용하여 block 형성. Block의 종류는 과제 pdf에 주어진 총 14가지 블록들이며, block_num은 random하게 주어짐 (interact_module에서 설명).

3-4) rotate()

Rotate의 경우 block에 사용하는 method로 불변 참조를 이용하여 90도 회전시킨 새로운 block을 이용함. 기존 block의 값을 변경하지 않은 이유는, board_module의 check_lose()에서 rotate()를 사용했을 때 현재 사용자의 블록에는 영향을 끼쳐서는 안되기 때문임. 따

라서 사용자가 실제 rotate를 한 경우 다시 대입하는 방식을 사용함. Rotate()를 위한 구현으로는 단순히 index를 이용함.

3-5) show()

Board_module()의 show와 같은 방식으로 구현된, block을 출력하기 위한 함수.

4. interact_module

4-1) get_instruction()

Command line을 통한 명령 즉, 0 혹은 좌표값을 입력 받기 위한 함수로, 해당 모듈 내의 interact 함수를 위해 사용됨. 유효한 입력이 주어진 경우 Some((i8,i8))을 반환하고, 아닌 경우 None을 반환함. 입력받은 문자열을 .split_whitespace()를 통해 iterator를 생성한 뒤 접근, 첫번째 값이 0이라면, Some((0,-1))을 반환함. Return type의 통일을 위해 (0,-1)의 -1을 집어넣은 것으로, 해당 경우에서는 -1은 아무런 의미도 갖지 않음. 만일 유효한 좌표, 즉 (1,1) ~ (7,7)의 경우 해당 튜플을 Some으로 감싸 반환하지만 아닌 경우 모두 None 반환.

4-2) interact()

Interact_module의 핵심 기능을 구현하는 부분. Board 인스턴스와 block 인스턴스, player 인스턴스가 주어진 상태에서 사용자의 입력을 get_instruction()을 통해 받은 값이 유효할 때까지 loop를 진행. 따라서 실수로 유효하지 않은 입력해도 다시 입력할 수 있도록 get_instruction() 반복. 해당 부분에서 print!() 매크로를 사용했을 때, 명령어 위치에 비해 출력이 나중에 되는 상황이 발생했지만, stdout에서 개행 여부를 통해 버퍼를 flush하게 되는데, 이 때문에 println! 매크로의 경우는 바로 출력이 되지만, 개행을 포함하지 않는 print!의 경우 출력 타이밍이 맞지 않을 수 있어 io::stdout().flush() 사용.

유효한 입력 값 (i8,i8) 튜플을 받았을 경우, ins_res.0 값이 0인 경우 rotate를 해야하고, 아닌 경우 해당 튜플을 좌표로 활용하여 block을 put해야함. Rotate하는 경우, 다시 사용자에게 input을 받아야 하는데, 이는 recursive하게 구현하였음. Put하는 경우도, 만일 유효하지 않은, 넣을 수 없는 위치에 put하게 되면 put_block의 결과가 None이므로, 불가능하다는 메시지를 출력한 후 다시 interact를 진행하도록 recursive하게 구현하였음. 만일 put이 성공적으로 진행되었을 경우 아무런 작업을 더 이상 진행하지 않고 interact를 마무리함.

4-3) gaming()

Gaming()은 main.rs에서 호출하는 유일한 함수로, 사실상의 main 함수 역할을 하는 함수임. Board를 초기화하고, end_condition도 초기화함. Loop 내부는 player의 한 턴을 의미함. 이 턴마다 유저에게는 랜덤한 블록이 주어져야하므로 rand crate를 활용하여 (toml 파일에 dependency추가) 0..=13, 즉 가능한 block_num를 생성한 후 get_block을 통해 사용할

block을 받게 됨. 그 후 interact()를 호출하여 사용자가 command line을 통해 상호작용 할 수 있음. Interact()가 특정 player의 한 턴을 맡는 함수라면, gaming()의 경우는 게임이 종료될 때 까지, 즉, board_module을 통해 check_lose가 true를 반환하는 경우 loop를 종료하게 됨. 만일 종료 조건이 달성되지 않았다면 (넣을 수 있는 위치가 있다면) change_player를 통해 player를 전환함. 종료 조건이 달성된 시점의 cur_player가 패배한 플레이어임.

4-4) change_player()

Gaming()에서 사용되는 함수로, 단순히 플레이어의 턴을 바꾸는 역할.

5. tests

5-1) test_get_block()

Get_block()이 잘 작동하는지 테스트 하는 함수, 0~13까지의 block을 모두 show, 출력하도록 함. Cargo test -- --show-output을 통해 제대로 block이 출력됨을 확인했음.

5-2) test_init_show()

Board의 init과 show()가 잘 동작하는 지 확인하는 함수. 마찬가지로 출력을 통해 제대로 동작하는지 확인함.

5-3) test_put_block()

Put_block이 잘 동작하는지 확인하는 함수로, 마찬가지로 show를 통해 제대로 put이 되었는지 확인하고, return 값이 적절한지 확인함.

5-4) test_check_lose()

Check_lose()가 잘 동작하는지 확인하는 함수로, board가 꽉찬 상태와 초기 상태인 경우의 return값을 각각 확인함.

5-5) test_change_player()

Change_player()가 잘 동작하는지 확인 하는 함수로, player change가 잘 일어나는지 cur_player를 통해 확인함.

```
problem_1 — -zsh — 157×67
Last login: Fri Nov 15 22:22:26 on ttys000
minuk-0815@minukair ~ % cd Lecture/SystemProgramming/Assignment2/problem_1
minuk-0815@minukair problem_1 % cargo test
   Compiling problem_1 v0.1.0 (/Users/minuk-0815/Lecture/SystemProgramming/Assignment2/problem_1)
warning: unused variable: `rot_block`
  --> src/lib.rs:85:21
   |
85 |                 let rot_block = &(rot_block.rotate());
   |                     ^^^^^^^^^ help: if this is intentional, prefix it with an underscore: `_rot_block`
   |
   = note: `#[warn(unused_variables)]` on by default

warning: variable does not need to be mutable
   --> src/lib.rs:196:17
    |
196 |             let mut break_condition:bool;
    |                 ----^^^^^^^^^^^^^^^
    |                 |
    |                 help: remove this `mut`
    |
    = note: `#[warn(unused_mut)]` on by default

warning: `problem_1` (lib) generated 2 warnings (run `cargo fix --lib -p problem_1` to apply 1 suggestion)
warning: `problem_1` (lib test) generated 2 warnings (2 duplicates)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.62s
     Running unittests src/lib.rs (target/debug/deps/problem_1-559e091e95e67e16)

running 6 tests
test tests::test_check_lose ... ok
test tests::test_get_block ... ok
test tests::test_change_player ... ok
test tests::test_init_show ... ok
test tests::test_rotate ... ok
test tests::test_put_block ... ok

test result: ok. 6 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests src/main.rs (target/debug/deps/problem_1-3e9425b99026c49c)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests problem_1

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

minuk-0815@minukair problem_1 %
```

```
P2's block:
 _ _ _
|@|@|@|
|_|_|@|
|_|_|_|

Put your block (r c) or Rotate (0): 0
P2's block:
 _ _ _
|@|@|_|
|@|_|_|
|@|_|_|

Put your block (r c) or Rotate (0): 3 3
   1 2 3 4 5 6 7 8 9
  _ _ _ _ _ _ _ _ _
1 |0|0|0|0|0|0|@|@|@|
2 |@|0|0|0|0|0|0|0|_|
3 |@|@|@|@|_|@|@|@|0|
4 |@|_|@|@|@|@|@|@|0|
5 |_|_|@|0|@|@|@|0|0|
6 |0|0|0|0|0|@|@|@|_|
7 |@|0|0|0|0|0|_|@|@|
8 |@|_|_|0|0|_|_|_|_|
9 |@|@|_|0|_|_|_|_|_|

P1's block:
 _ _ _
|_|0|0|
|0|0|_|
|0|_|_|

Put your block (r c) or Rotate (0): 0
P1's block:
 _ _ _
|0|_|_|
|0|0|_|
|_|0|0|

Put your block (r c) or Rotate (0): 0
P1's block:
 _ _ _
|_|_|0|
|_|0|0|
|0|0|_|

Put your block (r c) or Rotate (0): 7 5
   1 2 3 4 5 6 7 8 9
  _ _ _ _ _ _ _ _ _
1 |0|0|0|0|0|0|@|@|@|
2 |@|0|0|0|0|0|0|0|_|
3 |@|@|@|@|_|@|@|@|0|
4 |@|_|@|@|@|@|@|@|0|
5 |_|_|@|0|@|@|@|0|0|
6 |0|0|0|0|0|@|@|@|_|
7 |@|0|0|0|0|0|0|@|@|
8 |@|_|_|0|0|0|0|_|_|
9 |@|@|_|0|0|0|_|_|_|

P2's block:
 _ _ _
|@|@|@|
|_|_|@|
|_|_|_|

P2 fails to put the block. P1 wins!! congratulation!
```